

# Coding Dojo



# Qui suis-je ?

Christelle, développeuse et crafteuse Java.

 <http://www.linkedin.com/in/christelle-prut>

 <http://github.com/lifelightdev>

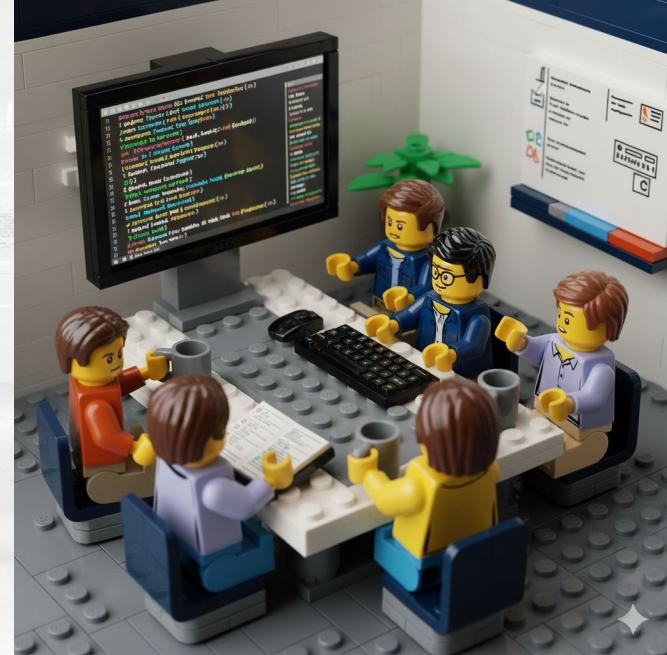
 <https://lifelightdev.blogspot.com>



Quel est le point commun entre  
un athlète de haut niveau  
comme Teddy Riner et un développeur ?



# L'entraînement



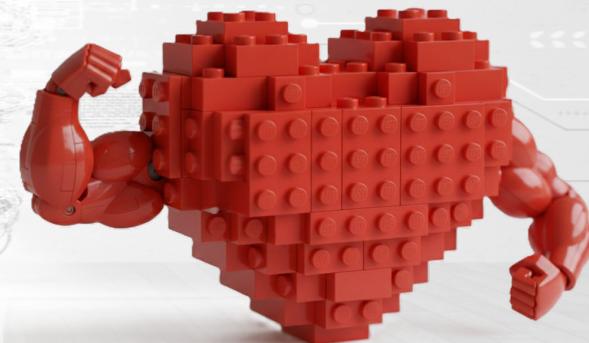
# Pourquoi s'entraîner ?

- Apprendre et s'améliorer



# Pourquoi s'entraîner ?

- Apprendre et s'améliorer
- Découvrir et expérimenter



# Pourquoi s'entraîner ?

- Apprendre et s'améliorer
- Découvrir et expérimenter
- Faire partie d'une communauté

















# Quelques définitions

Un kata



# Quelques définitions

Un kata

Un dojo



# Quelques définitions

Un kata

Un dojo

Un manifeste



# Quelques définitions

Un kata

Un dojo

Un manifeste

Le manifeste Agile



# Quelques définitions

Un kata

Un dojo

Un manifeste

Le manifeste Agile

Le manifeste Software Craftsmanship



# Fizz Buzz



# Fizz Buzz

Règle : Divisible par 3 = Fizz et divisible par 5 = Buzz



# Fizz Buzz

Règle : Divisible par 3 = Fizz et divisible par 5 = Buzz

Exemple : 1, 2, Fizz, 4, Buzz, Fizz, 7, 8 ...



# Fizz Buzz

Règle : Divisible par 3 = Fizz et divisible par 5 = Buzz

Exemple : 1, 2, Fizz, 4, Buzz, Fizz, 7, 8 ...

Méthodologie ? TDD, TCR ou TCRDD ...?



# Exemples de méthodologie

- Faire du TDD (Test Driven Development)

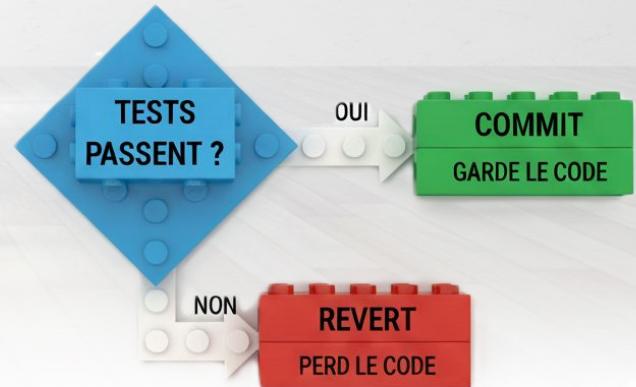


# Exemples de méthodologie

- Faire du TDD (Test Driven Development)



- Faire du TCR (Test && Commit || Revert).



# Fizz Buzz

Règle : Divisible par 3 = Fizz et divisible par 5 = Buzz

Exemple : 1, 2, Fizz, 4, Buzz, Fizz, 7, 8...

Méthodologie : TDD

Organisation : Mob programming, fishbowl ou randori ...?



# Le mob programming

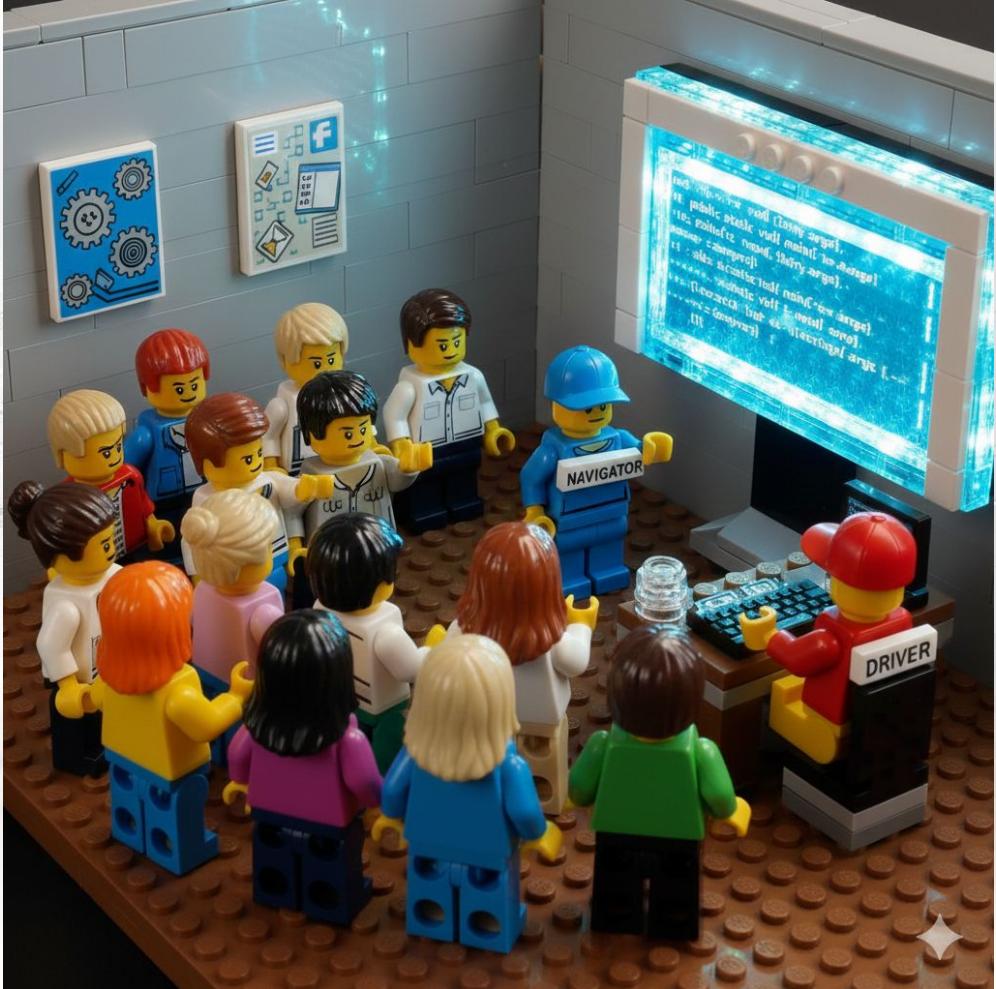
## Les rôles



# Le mob programming

## Les rôles

- L'assemblée



# Le mob programming

## Les rôles

- L'assemblée
- Le navigator



# Le mob programming

## Les rôles

- L'assemblée
- Le navigateur
- Le driver



# Exemples d'organisation

- Le fishbowl



# Exemples d'organisation

- Le fishbowl
- Le randori



# Fizz Buzz

Règle : Divisible par 3 = Fizz et divisible par 5 = Buzz

Exemple : 1, 2, Fizz, 4, Buzz, Fizz, 7, 8 ...

Méthodologie : TDD

Organisation : Randori



# C'est parti



# C'est parti

Quel est le plus petit test possible pour répondre à l'instituteur ?



# C'est parti

Quel est le plus petit test possible pour répondre à l'instituteur ?

$1 = 1$



The screenshot shows a Java IDE interface with the following details:

- Title Bar:** FB FizzBuzz > Version Control
- File:** FizzBuzzTest.java
- Code Content:**

```
1 package life.light;
2
3 > import ...
4
5
6 class FizzBuzzTest {
7     @Test
8     void un_egal_un() {
9         FizzBuzz fizzBuzz = new FizzBuzz();
10        String resultat = fizzBuzz.convert(1);
11        assertThat(resultat).isEqualTo(expected: "1");
12    }
13 }
```
- Toolbars and Icons:** Standard Java IDE icons for file operations, search, and help.
- Status Bar:** FizzBuzz > src > test > java > life > light > FizzBuzzTest
- Bottom Right Corner:** A small 3D model of a person working at a desk with a computer monitor.

The screenshot shows a Java development environment with two files open:

- FizzBuzzTest.java**:

```
package life.light;

public class FizzBuzz { 2 usages

    public String convert(int i) { 1 usage
        return null;
    }
}
```
- FizzBuzz.java**:

```
package life.light;

public class FizzBuzz { 2 usages

    public String convert(int i) { 1 usage
        return null;
    }
}
```

The code in both files is identical, defining a class `FizzBuzz` with a single method `convert` that returns `null`. A tooltip indicates there are 2 usages of the class and 1 usage of the method. A warning icon (yellow triangle) is shown in the status bar.

Bottom navigation bar:  
FizzBuzz > src > main > java > life > light > FizzBuzz

Status bar: 8:2 LF UTF-8 4 spaces

The screenshot shows a Java development environment with the following details:

- Project Structure:** FB FizzBuzz (selected), Version Control.
- Code Editor:** FizzBuzzTest.java (highlighted) contains a single test method: `void un_equal_un()`. The code uses the `assertThat` library to check if the result of `fizzBuzz.convert(1)` equals "1".
- Run Tab:** Shows the results of the run. One test failed: `FizzBuzzTest (life.light) 76 ms`, specifically the `un_equal_un()` test, which took 76ms. The error message is:

```
org.opentest4j.AssertionFailedError:  
expected: "1"  
but was: null  
Expected :"1"  
Actual   :null  
<Click to see difference>
```
- Bottom Status Bar:** FizzBuzz > src > test > java > life > light > FizzBuzzTest. Status: 6:7 CRLF UTF-8 4 spaces.

Oups,  
c'est déjà à la personne suivante



# C'est parti



# C'est parti

Mais que dois-je faire ?



# C'est parti

Mais que dois-je faire ?

Passer le teste au vert le plus simplement possible.



# C'est parti

Mais que dois-je faire ?

Passer le teste au vert le plus simplement possible.

Mais comment ?



# C'est parti

Mais que dois-je faire ?

Passer le teste au vert le plus simplement possible.

Mais comment ?

En retournant 1



The screenshot shows a Java development environment with two files open:

- FizzBuzzTest.java**:

```
package life.light;

public class FizzBuzz { 2 usages

    public String convert(int i) { 1 usage
        return "1";
    }
}
```
- FizzBuzz.java**:

```
package life.light;

public class FizzBuzz { 2 usages

    public String convert(int i) { 1 usage
        return "1";
    }
}
```

The code in **FizzBuzz.java** is identical to the test file. It contains a single method `convert` that returns the string "1" for any integer input. There is one warning icon in the status bar, indicating a potential issue.

Bottom navigation bar:  
FB FizzBuzz > Version Control > FizzBuzzTest > FizzBuzz > convert

Status bar: 7:6 LF UTF-8 4 spaces

The screenshot shows a Java development environment with the following details:

- Project Structure:** FB FizzBuzz
- Code Editor:** FizzBuzzTest.java (selected)
- Code Content:**

```
1 package life.light;
2
3 > import ...
4
5
6 class FizzBuzzTest {
7     @Test
8     void un_egal_un() {
9         FizzBuzz fizzBuzz = new FizzBuzz();
10        String resultat = fizzBuzz.convert(i: 1);
11        assertThat(resultat).isEqualTo(expected: "1");
12    }
13 }
```
- Run Tab:** Run FizzBuzzTest
- Run Results:**
  - FizzBuzzTest (life.light) 71ms
  - ✓ 1 test passed 1 test total, 71 ms
  - ✓ un\_egal\_un() 71ms
  - "C:\Program Files\Java\jdk-25\bin\java.exe" ...
  - Process finished with exit code 0
- Bottom Status Bar:** FizzBuzz > src > test > java > life > light > FizzBuzzTest | 6:7 CRLF UTF-8 4 spaces

# Les astuces



# Les astuces

- Avant de commencer :
  - Avoir un environnement de développement prêt à l'emploi.



# Les astuces

- Avant de commencer :

- Avoir un environnement de développement prêt à l'emploi.
- Avoir une personne qui connaît bien le langage.



# Les astuces

- Avant de commencer :
  - Avoir un environnement de développement prêt à l'emploi.
  - Avoir une personne qui connaît bien le langage.
- Au début :
  - Vérifier que l'affichage est lisible pour tous.



# Les astuces

- Avant de commencer :
  - Avoir un environnement de développement prêt à l'emploi.
  - Avoir une personne qui connaît bien le langage.
- Au début :
  - Vérifier que l'affichage est lisible pour tous.
  - Faire un rappel du fonctionnement.



# Les astuces

- Avant de commencer :
  - Avoir un environnement de développement prêt à l'emploi.
  - Avoir une personne qui connaît bien le langage.
- Au début :
  - Vérifier que l'affichage est lisible pour tous.
  - Faire un rappel du fonctionnement.
- Au milieu
  - Faire une orientation.



# Les astuces

- Avant de commencer :
  - Avoir un environnement de développement prêt à l'emploi.
  - Avoir une personne qui connaît bien le langage.
- Au début :
  - Vérifier que l'affichage est lisible pour tous.
  - Faire un rappel du fonctionnement.
- Au milieu
  - Faire une orientation.
- A la fin :
  - Faire une rétrospective.



# Les valeurs

- Zéro pression
  - Code éphémère



# Les valeurs

- Zéro pression
  - Code éphémère
  - Niveau sans importance



# Les valeurs

- Zéro pression.
  - Code éphémère
  - Niveau sans importance
  - Toutes les questions sont bonnes à être posées



# Les valeurs

- Zéro pression.
  - Code éphémère
  - Niveau sans importance
  - Toutes les questions sont bonnes à être posées
- Respect
  - Esprit positif



# Les valeurs

- Zéro pression.
  - Code éphémère
  - Niveau sans importance
  - Toutes les questions sont bonnes à être posées
- Respect
  - Esprit positif
  - Pas de jugement



# Les valeurs

- Zéro pression.
  - Code éphémère
  - Niveau sans importance
  - Toutes les questions sont bonnes à être posées
- Respect
  - Esprit positif
  - Pas de jugement
  - Le droit à l'erreur



# Outils

- Timer pour faire tourner les rôles du mob :  
<https://mobtime.hadrienmp.fr/>
- Système de vote : <https://emmanuelpaatz.com/dojosurvey>
- Site de kata : <https://codingdojo.org/kata/>
- Sandbox : <https://pinage404.gitlab.io/nix-sandboxes/>



# Où faire des dojos

- Dojo en présentiel à Paris :

[https://mobilizon.fr/@dojo\\_de\\_programmation\\_paris/events](https://mobilizon.fr/@dojo_de_programmation_paris/events)

- Dojo en distanciel avec les crafteurs de Lyon :

<https://www.meetup.com/fr-fr/software-craftsmanship-lyon/>

- Dojo en distanciel avec les crafteurs de Lille :

<https://www.meetup.com/software-craftsmanship-lille/>



# Craft

- Le manifeste du Craft

<https://manifesto.softwarecraftsmanship.org/#/fr-fr>

- Le mob programming :

<https://mobprogramming.org>

- Le Livre Extreme programming :

<https://www.eyrolles.com/Informatique/Livre/extreme-programming-9782744014338/>

- Le Livre Clean code :

<https://www.eyrolles.com/Informatique/Livre/coder-proprement-9782326002272/>



# L'important c'est le chemin parcouru plus que la destination



